Detecting Attacks in Computer Networks

Mike Fugate, James R. Gattiker
{fugate,gatt}@lanl.gov
Los Alamos National Laboratory

1 Introduction

This report describes work in classifying computer network traffic as normal versus network attacks. The data used is derived from an extended DARPA study that simulated a network with traffic intended to be typical of a military base[2]. This data was made into a standard and well explored reference when it was offered as the object of the KDD99 Conference "KDD Cup" contest[3]. The original DARPA data was reduced, and a feature extraction process was performed by domain experts[5]. Details of the dataset and experiment are discussed in Section 2.

This dataset and classification problem has been explored using many of the techniques in the data mining and machine learning toolbox[1][4], including classification trees[7], expert systems[9], nearest neighbor classification, and model ensembles. Our approach is to explore the dataset using the relatively new approach of Support Vector Machines (SVMs) against published benchmark results, and some benchmark results performed in this study to allow detailed analysis of results. Extensive reviews of SVMs are available [6]. In overview, SVMs are (in one instantiation) a method of separating a dataset into two categories. An interesting feature of the SVM approach is that it allows solution to the global optimization of the separation performance in a possibly nonlinear high dimensional mapping of the problem space. Other nonlinear methods such as backpropagation networks or decision trees provide no guarantees about optimal performance. On the other hand, this property of SVMs do not alleviate the issues of understanding the dataset, choosing appropriate models and model parameters. They introduce a requirement for skill in optimization theory to properly implement. Also, they rely on optimization that may require significant computation. On the balance, SVMs are widely considered to be a significant extension of capability in machine learning and many classic problems have been addressed, with excellent results.

This report will present results from different models optimized using SVMs, a method of ensemble ANOVA models, and the linear perceptron pocket algorithm. In addition to these benchmark models that we used for comparison of performance, we compare our results to those of published analysis of both the original DARPA and the derived KDD datasets. We discuss the data analysis and categorization.

Table 1: Attack distribution

Attack type	Training set %	Test set %
Normal	19.69	19.48
DOS	79.24	73.90
Probe	0.83	1.34
U2R	0.01	0.07
R2L	0.23	5.20

This report does not attempt to address issues in dataset generation or feature selection. In our view the details of the network and data collection process as well as the way in which this "raw data" is transformed into well-defined feature vectors is a very important problem. However that exploration is beyond the scope of this effort.

2 Dataset Description

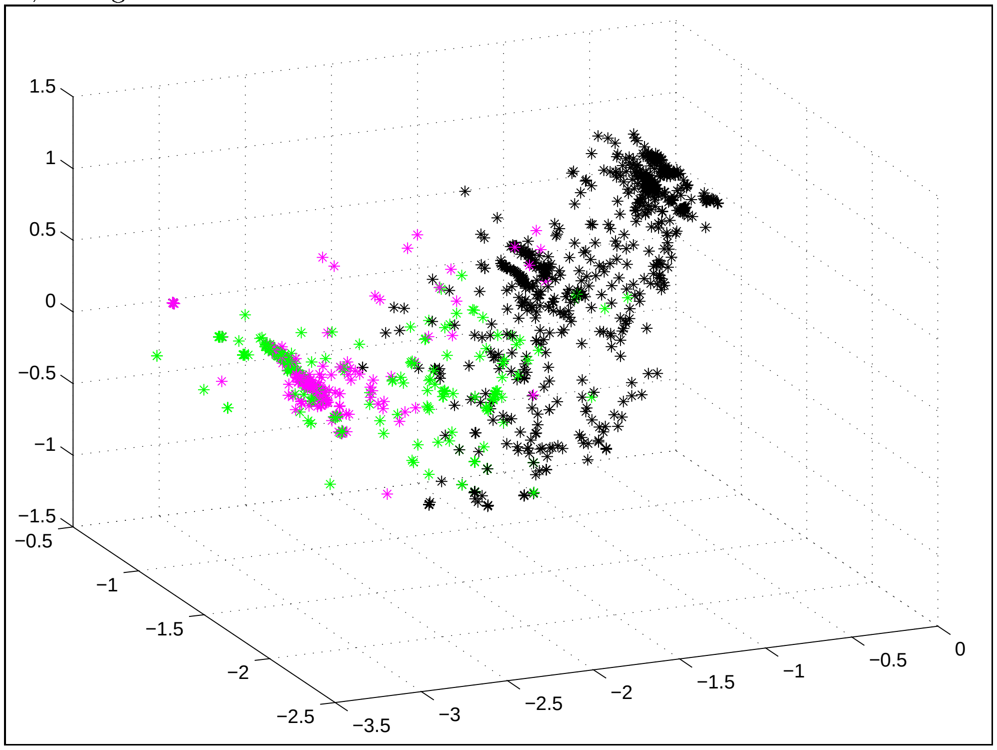
The resulting dataset has the following characteristics:

- 41 features derived from network packet headers,
- training and testing segments that are from consecutive weeks of test,
- a large dataset of 5 million training examples and 311,029 testing examples,
- a 10% sample of the 5 million training examples,
- several specific network attacks in four categories,
- data is labeled (as normal or by attack type)
- testing data is somewhat different in character (types of attack) from training data.
- raw data is transformed into scalar likelihoods for all variables by building ANOVA models on discrete values or intervals of continuous variables (see the discussion in Section 4.2)

The KDD Cup data with a complete description are available from the UCI data mining repository [8]. The attack distribution for the training and test data sets is given in table 1.

Figure 1 illustrates the different character, or *nonstationarity* of the training dataset compared to the test dataset. Black points represent non-attacks from both datasets. Red points are attacks from the training set, and green points are attacks from the test set. The data was randomly sampled for this visualization. The axes are the first three principal components of the sample from the training set. This clearly shows that, although some attack points remain in the same areas between the two datasets, many do not. This difference in distribution is a source of error that will not be overcome by static classification methods.

Figure 1: Distribution of dataset. Black is non-attack instances, red are attacks from the training set, and green are attacks from the test set.



3 Modeling Techniques

3.1 DARPA Data Collection Experiment

In 1998 DARPA in conjunction with MIT Lincoln Laboratory conducted an evaluation of computer intrusion detection systems. More information about this evaluation and much of the information in the following sections is taken from [2]. An intrusion detection evaluation test bed was developed which generated normal traffic similar to that on a government site containing 100's of users on 1000's of hosts. More than 300 instances of 38 different automated attacks were launched against victim UNIX hosts in seven weeks of training data and 2 weeks of test data. It is important to keep in mind that the test data had examples of attacks that were not in the training data and the distribution of attacks in the test data was different from that of the training data.

3.2 DARPA Performance summary

Six research groups participated in a blind evaluation and results were analyzed for the four attack types: probe, denial-of-service (DOS), remote-to-local (R2L), and user-to-root (U2R). The best systems detected old attacks included in the training data, at moderate detections rates (or true positive rates) ranging from 63% to 93%, depending on the type, at a false positive rate of 10 false alarms per day. Ten false alarms per day corresponds to a false positive rate of approximately 0.015%, and is seen as an acceptable workload for analysis of all alarms by a system security officer. Detection rates were much worse for new and novel R2L and DOS attacks included only in the test data. The best systems failed to detect roughly half of these new attacks. This illustrates that although detection of attacks may be possible at acceptable levels of performance for attacks that are very similar to previous attacks, new attacks are substantially more difficult to detect. Understanding how to detect new attacks is the direction that research in this area must ultimately take.

There are two issues standing in the way of comparing performance between the DARPA and the KDD-related explorations (including those in this paper). First, the KDD dataset consists of features derived from the DARPA data, but not precisely the same data. DARPA participants created these features with the intent that they capture the full information and then made the data publicly available. The second issue is that the DARPA results are computed on *attack groups*, while the KDD results are computed on *attack instances*. The distinction is that there may be many attack instances in an attack group. For the DARPA study, an attack group is detected if a single communications packet is flagged. The KDD dataset does not contain information about which attack is in which group, and so the evaluation cannot be made on this basis. The same model should perform much better when the performance is measured in attack groups than when it is measured as independent packets.

3.3 DARPA Results by Attack Type

We now proceed with a summary of performance by attack. For DOS attacks, the best system, an expert system ("DARPA Expert" in Table 2), detects 65% of the attacks with just one false alarm per day. The pattern recognition system ("DARPA PR"in Table 2) detected slightly more than 65% of the attacks but with a false alarm rate of about 7 per day.

Probe attacks were the easiest attacks to detect and most of the systems did very well in this category. Two expert systems detected nearly 90% of these attacks with 1 false alarm per day and the pattern recognition system detected almost 97% of the attacks at a false alarm rate of about 7 per day.

In the U2R category the pattern recognition system performed best and detected roughly 70% of the attacks at a false alarm rate of less than 4 per day.

R2L attacks were the most difficult to detect for all systems. The best performing system was an expert system that detected fewer than 35% of the attacks at a false alarm rate of about 7 per day. The pattern recognition system had a detection rate of about 20% at 3 false alarms per day.

Overall, the best system, an expert system, detects roughly 75% of all attacks in the test data with fewer than 2 false alarms per day (0.003%) . The next best was the pattern recognition system and it detected 65% of the attacks with 20 false alarms per day (0.03%).

The best performing systems detect old attacks with reasonable accuracy ranging from 63% to 93% at 10 false alarms per day. Performance is much worse for new attacks. Detection accuracy for new attacks is below 25% for the R2L and DOS categories and not significantly different from performance with old attacks for the other two categories.

3.4 KDD results

The pattern recognition system used in the 1998 DARPA study was developed by Professor Sal Stolfo of Columbia University and Professor Wenke Lee of North Carolina State University. Stolfo and Lee made available a version of the DARPA data for use in the KDD99 Conference KDD Cup. The training data consists of 5 million records and each record contained 41 features derived from network packet headers. Associated with each record is a label to identify the attack type. The test data (not available to the KDD competitors) has 311,029 examples.

The goal for KDD Cup participants was to build a multi-class predictive model capable of distinguishing between legitimate and illegitimate connections in a computer network. Entries were ranked according to the average cost per test example, with the winning entry having the smallest average cost. In this evaluation each packet was scored independently rather than in attack groups. A complete description of the task, features, and results can be found in [3].

When viewed as a two-class predictive model the KDD Cup winning entry had an overall misclassification error rate of 6.7%; the true positive rate was 91.8% and the false positive rate was 0.55%.

4 Benchmark Methods

This section describes the methods that were directly applied to the data for this study. The previous work shows benchmark overall results, but not specific data classification records that can be used for true comparative analysis of performance. It is important to understand whether methods from different model families are scoring examples similarly, even if the overall results are the same. We have chosen, as discussed above, the highly nonlinear models of Support Vector Machines, the linear Perceptron trained with the pocket algorithm, and our method of ANOVA ensembles which can be thought of as a piecewise uniform model. These have all been demonstrated to have good performance in large data modeling domains.

4.1 Support Vector Machines

Boser, Guyon, and Vapnik [10] proposed a two-class classifier, called the support vector machine, that finds a hyperplane which maximizes the minimum (signed) distance between the plane and the training points. For training data x, the equation of the hyperplane is

$f(x) = w^T x + b$. In general x is a p-dimensional feature vector and the T on w indicates the transpose of a vector. The normal vector to the hyperplane is w and b is know as the bias or offset. Both w and b must be estimated from the training data. In most problems only a subset of the training points (i.e., nearest the hyperplane) are needed to construct a linear estimate of w. The particular training points that are used to estimate w are referred to as "support vectors". See Vapnik [11] for a more complete discussion of support vector machines for classification.

For a two-class problem with labels of -1 for class 0 and 1 for class 1, a new feature vector, z, is assigned to class 1 if $w^T z + b > 0$ otherwise it is assigned to class 0. Here we are assuming that w and b have been estimated from the training data. The number 0 in this classification rule is referred to as the threshold. By changing the value of the threshold the number of observations that are assigned to each class can be changed and thus the detection rate and the false positive rate can be changed to some specified values.

In recent years support vector machines have shown promising results on many real-world problems. In addition, they go hand in hand with Vapnik's structural risk minimization (SRM) principle. Vapnik's SRM allows one to obtain *distribution independent* probability bounds on the generalization error of a classifier [11]. Note that these probability bounds may be very loose and could even be larger than 1.0.

This study used the public domain SVM-Light package, which is capable of building SVM models with various parameters[12].

4.2 ANOVA Ensemble Model

The ANOVA ensemble model was developed to address a large dataset composed of primarily categorical data; if continuous variables are also present these are transformed into categorical variables by defining intervals on continuous data. Categorical variables and intervals of continuous variables are transformed into corresponding probabilities of positive category membership by counting examples from the training data, and these probabilities are then combined using a linear model to optimize performance. Because of the final regression integration stage, this is similar to the "naive Bayes" approach, which multiplies the estimated posterior probability from multiple independent marginal estimates. The ANOVA ensembles has been shown to give very good performance on large classification problems with mixed categorical and continuous indicators, without requiring extensive tuning or data exploration.

One important characteristic of this method is a guaranteed convergence with a fixed number of passes through the data. The data must be examined (in a pass) once in the interval calculation, again to do counting statistics, again to do replacement of ranges with probabilities, and twice in the linear regression. Additional overhead are the requirement to sort the data columns individually, and to do the matrix inversion (a trivial operation in comparison). This describes generating intervals of equal population; if the data is characterized intervals may be set in other ways eliminating the sort operation. Overall, this is a relatively fast algorithm with explicitly bounded computation time.

The combination of these values is a simple regression between these likelihoods and the independent variable (0 for normal, 1 for attack). However, the regression matrix inverse uses only the first five factors, which has been empirically demonstrated to make the method less sensitive to noise in the less important variables.

4.3 Perceptron Pocket Training

The perceptron pocket training method arrives at a linear separation of data in the input space, and is applicable to multi-class problems as well as the two-class problem. It has an advantage over some *ad-hoc* perceptron training methods, in that it is guaranteed to converge, although the guarantee is that it will converge in a finite but possibly very long time, and without stopping criteria [9]. The characteristics of this method are that it is an iterated algorithm that does not bound the number of operations to convergence or good performance. Empirically, we have found this method to deliver a competitive linear classifier in a reasonable training time. In this case, we used the result after one million iterations.

4.4 Model-Building Performance

The aim of this study was not to compare the speed of model building and scoring. On this dataset, all of the methods of interest could be computed on a desktop system in a reasonable time. One possible exception is that the SVM method was not computable in reasonable time using the entire training set. Our experience was that there was negligible penalty in training on the smaller sample as we have. This was established by working with different training set sizes in the neighborhood of the training set used and noting that no performance degradation was evident.

The ANOVA method computes the model parameters in a timescale of seconds to minutes, the perceptron pocket training is arbitrary but for our chosen number of iterations was in the range of an hour. The SVM training ranged from minutes to hours nominally, with some issues with non-convergence using apparently difficult parameter settings. It is not clear whether this was true non-convergence or just exhausting our patience.

5 Classification Performance Comparison

5.1 Evaluation Methodology

The key observation about the data is that the distribution of the supplied training and test datasets is not the same. This *nonstationarity* is key to the problem, however it may not be informative in model evaluation. The training set is assumed to be drawn from a single distribution. Therefore, we split the training set in order to evaluate model performance on the problem with stationary data, as well as on the nonstationary test set. Our view is that the nonstationary test set has an inherent upper limit in performance that cannot be exceeded with any classification method trained on the training set, because information is not available in the training set to predict the nonstationarity. Thus it is important to consider performance on stationary and nonstationary test sets in model comparison.

The training dataset has been split into 3 parts by random sampling, that we will denote *training*, *tuning*, and *validation*. We refer to the test set as *test*. The training set is used to estimate model parameters; the tuning set is used for model parameter setting in iterated model selection; the validation set is used to evaluate performance on the stationary data; and the test set is used to evaluate performance on nonstationary data. The train data set has 49,305 examples; the tune data set has 11,197 examples; and the validation set has 433,518 examples.

In this problem it is useful to consider performance on subclasses of the category of attack. These subclasses are computed from dataset descriptions in the literature treating the KDD dataset rather than being directly supplied with the data. We cannot be sure that the translation to the 4-class attack types is identical with previous studies, due to vagueness and contradiction in the published information. The subclasses are not homogeneous with respect to classification performance or their characteristics of nonstationarity. By considering the context of these subclasses model performance can be usefully interpreted in the application, although this interpretation is necessarily qualitative.

The performance is captured in the two features *detection rate*, defined as the proportion of attacks classified as attacks; and *false-positive rate*, defined as the proportion of normals classified as attacks.

For a given classification rule, by changing the threshold we can change the detection and false-positive rate for that classifier. It is debatable whether this is always valid. It is defensible in the case of linear methods such as ANOVA and Perceptrons, where a linear model can be seen as a smooth continuum from one class to the other. However, the SVM method can be highly nonlinear in the original problem space. Whether a model that was constructed for an optimal classification performance according to a specific criteria has any validity when moved in a narrow neighborhood has any validity is questionable. Nonetheless, in order to explore tradeoffs we have used this method to explore the space. The alternative would be to use a weighting scheme to build distinct models for each performance point, but the performance point is not easily specified without extensive parameter exploration. In the interest of time we resorted to the former method.

5.2 Evaluation Limitations

Performance can be difficult to summarize for a number of reasons. One fundamental limitation is that classification performance is not a scalar, and does not have an order. Thus a choice must be supplied about where to make a tradeoff in performance if an ordering is to be achieved.

The best overall error may not be the most desirable performance point in the context of an application, which in this case is that a reasonable false positive rate must be achieved. A method used generally is the analysis of detection rate versus false positive rate. This results in an entire performance spectrum for each method.

We have chosen three performance neighborhoods for the results reported, in order to fairly summarize the results. Note that for some methods we were not able to determine values for some of the cells in the table; e.g. there was no information in the DARPA report on overall error rate for the detection systems used. Empty table cells imply we were not able to obtain the necessary information.

5.3 Results

Table 2 shows summary performance for the methods applied to the validation set. The overall error rate corresponds to the detection rate given under the column heading P1. There is no entry for the KDD winner because KDD Cup entries were only evaluated on the test set.

All methods perform very well with three support vector machines having overall error rates of about 0.08% at a detection rate near 99.9% . From column P3 we see that all of the support vector machines detect more attacks with fewer false positives than the best DARPA participant (even with the looser group evaluation paradigm adopted in the DARPA study).

Table 3 shows summary performance for the methods when applied to the test data. Recall that the test data has a different distribution than the training data. Not surprisingly, the performance of all the methods on the test data deteriorates when compared to performance on the validation data. The KDD winner has the lowest overall error rate, 6.7%, at a detection rate of 91.80% and a false positive rate of 0.55%. In Table 3 the overall error rate for each method corresponds to the detection rate given in column P1.

Even though the overall error rates are very similar for all methods, the false positive rate for the KDD winner is much better than what is achieved by the other techniques, at the same detection rate. This result is somewhat surprising considering that the support vector machines are solving a 2-category classification problem and the KDD winning entry was designed to minimize the average cost per example over 4 attack categories.

Exploring the performance results provides useful insight into the problem. In this dataset there is more information than a single-class discriminator, in fact there are 5 distinct categories (4 types of attack plus normal). As has been observed previously [2], it is useful to consider the performance by category on the test set of new attacks. A summary of these results is shown in Tables 4 and 5.

6 Discussion

The results indicate that competitive performance can be achieved in this classification problem using a number of different tools. These similar results suggest that these methods

Table 2: Validation set performance summary

Method	Overall	P1		P2		P3	
	error %	det%	fp%	det%	fp%	det%	fp%
DARPA		93.00	0.015				
SVM linear	0.17	99.89	0.39	98.52	0.046	97.88	0.007
SVM 2poly	0.08	99.93	0.11	98.51	0.011	97.87	0.002
SVM 3poly	0.08	99.94	0.15	98.51	0.014	97.88	0.003
SVM RBF	0.07	99.94	0.11	99.51	0.021		
ANOVA ens.	0.59	99.86	2.43	98.67	0.018	98.06	0.001
Pocket 2cl.	1.31	98.38	0.06	98.38	0.061	97.87	0.009
Pocket mcl.	0.20	99.89	0.56				

Table 3: Test set performance summary

Method	Overall	P1		P2		P3	
	error %	det%	fp%	det%	fp%	det%	fp%
DARPA Expert						75.00	0.003
DARPA PR						65.00	0.030
KDD Winner	6.70	91.80	0.55				
SVM linear	6.89	91.83	1.62	89.32	0.21	74.50	0.074
SVM 2poly	6.95	91.79	1.74	89.33	0.26	75.11	0.058
SVM 3poly	7.10	91.67	1.94	89.32	0.25	75.25	0.097
SVM RBF	6.86	91.83	1.43	89.30	0.21		
ANOVA ens.	6.88	91.67	0.90	89.36	0.10	87.95	< 0.001
Pocket 2cl.	7.00	91.80	1.52	89.24	0.23	85.43	0.063
Pocket mcl.	6.93	91.86	1.96				

Table 4: Validation set performance by attack type

Method	Overall			DoS	Probe	R2L	U2R
	error%	det%	fp%	det%	det%	det%	det%
% attacks in category				98.7	1.0	0.28	0.014
SVM linear	0.17	99.89	0.39	99.98	98.21	79.11	10.00
SVM 2poly	0.08	99.93	0.11	99.99	98.40	88.27	2.00
SVM 3poly	0.08	99.94	0.15	99.99	98.43	91.35	6.00
SVM RBF	0.07	99.94	0.11	99.99	99.06	90.02	20.00
ANOVA ens.	1.07	98.67	0.02	99.18	77.65	0.93	2.00
Pocket 2cl.	1.31	98.4	0.06	99.33	35.71	3.81	2.00
Pocket mcl.	0.20	99.89	0.56	99.96	98.54	83.95	20.00

Table 5: Test set performance by attack type

Method	Overall			DoS	Probe	R2L	U2R
	error%	det%	fp%	det%	det%	det%	det%
% attacks in category				91.8	1.7	6.5	0.09
KDD Winner	6.70	91.80	0.55	97.69	87.73	10.26	26.32
SVM linear	6.89	91.83	1.62	97.38	81.76	16.55	21.93
SVM 2poly	6.95	91.79	1.74	97.41	86.44	14.74	1.75
SVM 3poly	7.10	91.67	1.94	97.62	88.45	9.35	2.63
SVM RBF	6.86	91.83	1.43	97.30	79.26	18.29	25.88
ANOVA ens.	6.88	91.67	0.90	97.64	87.52	8.51	53.94
Pocket 2cl.	6.90	91.80	1.52	97.40	85.84	14.77	29.82
Pocket mcl.	6.93	91.86	1.96	97.65	86.79	11.45	54.38

exploit much if not all of the information available for classification. This suggests that radically better performance is unlikely, although it is certainly not impossible that some method could be devised to perform better. It should be noted that the methods documented are all free of problem context (except the DARPA expert system). That is, they are not constrained by information known to be relevant to the problem (that is, beyond the context provided in feature construction and selection). At this point in a classification data mining problem, moving to systems that are specifically designed with prior information to structure the models is a reasonable direction.

It can be seen from this analysis that the majority of the individual attack packets are from the denial of service attacks, which is to be expected. These are caught easily by the systems. The ambiguity in a 2-class problem is whether this performance is due to the detection problem being inherently easier for this class, or if the larger number of training examples in these classes leads to an optimization of reducing total error that favors this consistent category to the detriment of performance on smaller categories of examples.

A potential weakness in the evaluation is that the attack costs are all the same. This is perhaps unrealistic. Although a denial of service attack is disruptive, it is also obvious and without long-term damage to the network. On the other hand, a user to root attack is potentially devastating, since an intruder has gained control of the system(s) and has access to all information. The weighted results could show a different picture, specifically much lower performance if the denial of service performance becomes a much smaller contribution to average performance. On the other hand, there are very few examples of these sort of attacks, so it is not clear whether the results on this dataset can be taken as a conclusive indicator.

An overall interpretation of the results is that detection of the denial of service and probe attack types is sufficiently accurate to consider relying on automated detection. However, the remote to local and user to root attacks do not have sufficient detection accuracy to rely on only the indicators from classifiers.

The results on these two categories are somewhat paradoxical; the performance on the validation set, which is drawn from the same data as the training set, is often worse than the performance on the test set which is documented to be nonstationarity with respect to the training set. We do not have any analysis to understand this at this time, except to say that due to the small numbers of examples involved this effect may be only noise and not indicative of a "true" performance for the attack types. Nonstationarity is very evident in the degradation of performance in the remote to local attack type, at least. Assessment of the specific impact and ways to address it will be the topic of further work.

Further work in this area should proceed in two areas. The first is the inclusion of anomaly detection methods to either classify or to address the nonstationarity discussed. This work is proceeding and will be presented in another report. The second area is the development of an on-line testbed that can be used to try methods. There is only so much that can be learned from static historical data, especially in this area where attack types and strategies change quickly. Methods that can function in an on-line environment, as well as methods that are on-line adaptive can only be explored in a testbed network. This testbed could either be a network set up specifically to detect intrusions, or a functional network system that is also used for these research purposes. The setup and management of a testbed environment is a significant effort. It would need to be monitored to such a degree that all attacks are identified with reasonable certainty. Implementation of on-line tools is a much higher degree of effort than those that treat a static dataset. Finally, paradigms for managing reports from the automated detection system are necessary. This provides many more interesting research directions, as well as making it possible to develop a system that can be truly considered an "alpha" level user package, but at a high cost for operations.

References

[1] David J. Marchette, *Computer Intrusion Detection and Network Monitoring: A Statistical Viewpoint*, Springer-Verlag, NY, 2001.

[2] Richarp P. Lippmann *et. al.*, "Evaluating Intrusion Detection Systems: The 1998 DARPA Off-line Intrusion Detection Evaluation", Proc of the DARPA Information Survivability Conf., vol. 2, pp. 12-26, 1999.

[3] Charles Elkan, "Results of the KDD'99 Classifier Learning Contest", ACM SIGKDD Newsletter, Vol. 1, No. 2, Jan. 2000.

[4] Trevor Hastie, Robert Tibshirani, Jerome Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, Springer-Verlag, 2001.

[5] Sal Stolfo, Wei Fan, Wenke Lee, Andreas Prodromidis, Philip K. Chan, "Cost-based Modeling for Fraud and Intrusion Dectection: Results from the JAM Project", DARPA Information Survivability Conference 2000.

[6] Nello Cristianini, John Shawe-Taylor, *Support Vector Machines, and other kernel-based learning methods*, Cambridge University Press, 2000.

[7] Leo Breiman, Jerome H. Friedman, Richard A. Olshen, Charles J. Stone *Classification and Regression Trees*, Chapman & Hall, 1984.

[8] *KDD Cup 1999 Data*, http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

[9] Stephen I. Gallant, "Neural Network Learning and Expert Systems", MIT Press, 1993.

[10] Boser, B., Guyon, I. and Vapnik, V. (1992). A training algorithm for optimal margin classifiers. In *Proceedings of COLT II*, Philadelphia, PA.

[11] Vapnik, V. (1996). *The Nature of Statistical Learning Theory.* Springer, New York.

[12] T. Joachims, "Making large-Scale SVM Learning Practical", *Advances in Kernel Methods - Support Vector Learning*, B. Schlkopf and C. Burges and A. Smola (ed.), MIT-Press, 1999.